REMARKS

Claims 1-4 are pending in the present application, and as amended, claims 1 and 2 are the independent claims.

In the Official Action, dated September 2, 2005, the present application was withdrawn from allowance in view of the following rejections. As reinstated, claims 1-4 stand rejected under 35 U.S.C. § 103 as allegedly obvious over U.S. Patent No. 5,872,973 (Mitchell) in view of U.S. Patent No. 6,690,761 (Lang). Claims 1-4 are also rejected under 35 U.S.C. § 101 for lacking a subject matter recitation tied to a computer readable medium. Claim 2 was objected to for depending improperly from claim 1. A request was made to update the related application section in the background section. Lastly, claims 1-4 were rejected under the judicially created doctrine of double patenting over claims 1-44 of U.S. Patent No. 6,684,246 B1 (parent of this divisional application). The outstanding rejection to the claims based on the art of record is respectfully traversed.

Formal Matters

In response to the Examiner's request to update the related application information, the background section of the present application has been updated herein to reflect the current information.

In consideration of the objection to the form of dependent claim 2, claim 2 has been rewritten in independent form to reflect the alternative nature of its previous recitation. Accordingly, withdrawal of the objection to the form of claim 2 is respectfully requested.

The Double Patenting Rejection

Claims 1-4 stand rejected under the judicially created doctrine of double patenting over claims 1-44 of U.S. Patent No. 6,684,246. Submitted on May 6, 2005 was a timely filed Terminal Disclaimer concerning U.S. Patent No. 6,684,246. Withdrawal of the rejection based on the judicially created doctrine of double patenting is thus respectfully requested in view of the previously filed Terminal Disclaimer.

The Rejection under Section 101

In consideration of the rejection to claims 1-4 under 35 U.S.C. § 101, claims 1 and 2 were amended to explicitly state they are limited to a data structure *stored on a computer readable medium*. Withdrawal of the rejection based on non-statutory subject matter is thus respectfully requested.

Summary of the Invention

Prior to Applicant's invention, to track access to a server object by client objects, server objects required the addition of code specifically directed to supporting the unique identification of client objects. Similarly, each client object needed to be programmed to receive and provide its unique identification.

Advantageously, with Applicant's invention, existing server objects can be used in such a way that each client object can be individually identified when it invokes a function. No modification to existing server class definitions and client class definitions is necessary.

Applicant's client tracking system provides a derived client tracking server class that includes an overriding implementation of a query function of the server class. The overriding implementation instantiates a phantom server object and returns a pointer to the instantiated phantom server object. The phantom server object has functions that correspond to and override the functions of the server class. These overriding functions perform custom processing on a client-by-client basis and forward their invocation to the corresponding functions of the server object. When a client invokes the query function of the client tracking server object, a pointer to a phantom server object is returned. From then on, when that client invokes a function of the phantom server object, custom processing can be performed for that client.

In one embodiment, the client tracking system specifies a phantom manager class for controlling the instantiation and destruction of the phantom server objects. The phantom manager class may provide a create function that, when invoked by the query function of the client tracking server object, instantiates a phantom server object and returns a pointer to the phantom server object. The phantom manager class may also provide a phantom going away function that, when invoked by a destructor of a phantom server object, performs custom processing upon destruction of a phantom server object. A developer who wants to track client accesses to a server class that is already defined may specify a derivation of the server class, referred to as a "client tracking server class." The developer may provide an overriding function of the query function as part of the client tracking server class. The developer may also provide an implementation of the create function and the phantom going away function of the phantom manager class that are tailored to the server class. In addition, the developer may provide implementations of the functions of the server class as part of the phantom server object. These implementations of functions of the phantom server object may perform the desired custom processing on a client-by-client basis.

Rejection under 35 U.S.C. § 103

Claims 1-4 stand rejected under 35 U.S.C. § 103 as allegedly obvious over Mitchell in view of Lang. Applicant respectfully traverses as follows. Without conceding the propriety of the substance, and given the heavy cautions against hindsight reconstruction of an invention, Applicant previously believed that the combination of (1) a system that generically provides for dynamically reconfigurable links (with transformations) between objects at runtime while reducing dependencies with (2) a system that accurately and reliably evaluates bone mineral density and bone structure from x-ray images under 35 U.S.C. 103 is absurd. That one of ordinary skill in the art would be led to combine these two references is beyond Applicant's imagination.

In this regard, as explained previously, there are three possible sources for a motivation to combine references: the nature of the problem to be solved, the teachings of the prior art, and the knowledge of persons of ordinary skill in the art. In re Rouffet, 149 F.3d 1350, 1357 (Fed. Cir. 1998).

Clearly, bone mineral density and bone structure measurements involve wholly separate problems from generically providing dynamically reconfigurable links between objects at runtime while reducing dependencies.

As to the teachings of the prior art, not surprisingly, Applicant cannot find any teaching or suggestion within Mitchell itself to combine Mitchell with Lang, or within Lang itself to combine Lang with Mitchell.

As to the knowledge of persons of ordinary skill in the art, Applicant respectfully submits that a person of ordinary skill in the software arts would not have knowledge of bone mineral density and bone structure measurements without consulting an expert doctor or two on the subject. Thus, clearly these two fields (two different specialties) include little overlapping knowledge.

For these reasons, Applicant respectfully submits that the previous rejection should remain withdrawn. Reconsideration based on the unrelatedness of the references is again requested.

Addressing the substance nonetheless,

Claim 1 recites:

...a client tracking server object derived from a server class that provides an implementation of a query interface function that overrides the query interface function of the server class...

This element of the claim therefore requires all of the following:

(1) A client tracking server object

(2) The client tracking server object is derived from a server class

(3) The server class from which the client tracking server object derives provides an implementation of a query interface function; and

(4) The implementation of the query interface function provided by the server class overrides the query interface function of the server class.

Applicant respectfully submits that the Official Action has failed to make a prima facie case of obviousness as to this collective set of elements of the claim. As best as Applicant can surmise, Mitchell was cited for its disclosure at the following places according to the underlining added by Applicant:

...the client object must be aware of the public interface (API) of the server object and thus dependencies are created just as in client server relationships between objects in the same program. **Col. 6, lines 5-22**

It is the express goal of the present invention to provide a method and system for generically providing dynamically reconfigurable links (with transformations) between objects at runtime without requiring those objects to become dependent upon, or have knowledge of each other.

It is also an objective to do this with minimal impact on the implementation of the classes being created. The architecture sought is a server/server architecture between objects, within a single program, between programs and or processes running on the same machine and between programs and or processes running on machines connected over a network. **Col. 6, lines 25-37**

1. When the left hand side's field (101) is changed, all of its probe callbacks are called.
2. The EosMapElement::leftHandSideCallback, a member of EosMapFieldToField (103) is called (this is the raw probe callback). Note that in an alternative embodiment this callback could be in the EosFieldElement (107).
3. A flag is set indicating that the mapper is active. (The flag is cleared after leftHandSideChanged returns)
4. The leftHandSideCallback then calls a virtual function of the mapper class called leftHandSideChanged.
5. The base class implementation of leftHandSideChanged calls a virtual function of the mapper called valueChanged. Most mappers override valueChanged rather than leftHandSideChanged in order to do transformations without regards to direction. The parameters are: The field that is changing, the type element that is changing (in this case the left side (107)), and the type element for the other side (104). The type element may represent a field, function, property or a list of members.

void EosMapElement::valueChanged(EosProbeObject *src,EosTypeElement *srcEl,EosTypeElement *destEl);

In the simple case of the field to field mapper, valueChanged simply gets the value from the source side parameter (that changed) and sets the value on the destination side to this new value through the right type element (104).

6. The right type element (104) sets the field (105) in the right hand object (106) through the dynamic binding mechanism. The type element has already precomputed the index of the field from the field name, so that this is as efficient as possible.
7. Probes now fire on the right hand field (105), which causes rightHandSideCallback of the mapper to be called. (Again, in an alternative

embodiment this could be in the field element (104) rather than directly in the mapper) This callback again checks to see if the mapper active flag is set, and since it is, processing stops; that is, the value is not forwarded to the left hand field (101). This check helps to avoid infinite looping within the mapping machinery. (The probes mechanism also has circular reduction machinery, but adding it to the mappers is more efficient and safer.) Of course this behavior can be overridden in cases where it makes sense to do so. **Col. 14, lines 10-56**

The path object also solves some really difficult situations related to the dynamic nature of objects in a running program. If, for example, a semantic link were made to a field inside of a pointer field, and the pointer field is reassigned to point to a new object, then the mapper has to disconnect its probes to the first object, and reconnect to the new object. The attachments to subfields of subfields is kept track of by keeping a path of field names, for example, fieldA.fieldB.fieldC would mean that fieldC is a member of fieldB, which is a member of fieldA. The path object plants probes on each of these intermediate levels and when an intermediate level changes, it calls the mapper's functions to reset their probes. Mappers must be able to detach from the old object and reattach themselves to the new objects as they change at runtime, but the path object does all the complex bookkeeping. The raw probe machinery is only aware of individual instances to probe, and does not have the contextual ability to detach and reattach as necessary like the mapper's and paths do. If probes are planted by hand, this functionality may not be obtained automatically. Dynamic binding and meta-data are used to find the new item to plant probes on. **Col. 16, line 60 to Col. 17, line 14**

```
// probe object server class
 class EOS.sub.-- MODIFIER EosProbeObjectServer : public EosProbeObject {
 public:
  EosProbeObjectServer();
  virtual .about.EosProbeObjectServer();
  // sender is EosConnection::connectProbeField
  virtual long setupServer(EosConnection *connection,long
```
Col. 36, lines 60 to 65

Most of the functionality of all of the type elements is accessed through the interface to EosTypeElement. That is, the functionality is exposed in virtual functions of the subclasses that is accessed, is for the most part, accessed through the common API of EosTypeElement. This makes each type work in a similar fashion when it makes sense to do so. **Col. 18, lines 16 to 22**

In short, none of these unrelated passages of Mitchell teach or suggest "a client tracking server object derived from a server class that provides an implementation of a query interface function that overrides the query interface function of the server class," as recited in claim 1 because none of those passages teach all of the following:

1: A client tracking server object

2: The client tracking server object is derived from a server class

3: The server class from which the client tracking server object derives provides an implementation of a query interface function; and

4: The implementation of the query interface function provided by the server class overrides the query interface function of the server class.

As stated in paragraph [0058] of the present application, Figure 11 is a flow diagram of the query interface function of the client tracking server class, corresponding to the query interface function of the IUnknown interface. This function is passed the identification of an interface and returns a pointer to the interface. The implementation of this function in the client tracking server class, which inherits the server class, overrides the implementation of this function in the server class.

In brief, Mitchell nowhere teaches or suggests these collective features of the invention, as recited in claim 1, no matter how many unrelated passages are strung together. Claim 2 recites a similar set of features that are nowhere taught or suggested by Mitchell. Claims 3-4 depend from claim 1 and are believed allowable for the same reasons.

Lang was cited for reasons related to the phantom manager class, but also does not cure at least the above-identified deficiency of root reference Mitchell with respect to Applicant's claims.

Accordingly, reconsideration and withdrawal of the rejection under 35 U.S.C. § 103 is respectfully requested.

CONCLUSION

Applicant believes that the present reply is responsive to each of the points raised by the Examiner in the Office Action, and submits that claims 1-4 of the application are in condition for allowance. Favorable consideration and passage to issue of the application at the Examiner's earliest convenience is earnestly solicited.

Date: December 2, 2005

[illegible]
Thomas E. Watson
Registration No. 43,243

Woodcock Washburn LLP
One Liberty Place - 46th Floor
Philadelphia PA 19103
Telephone: (215) 568-3100
Facsimile: (215) 568-3439